# SPECIFICATION

Electronic Version 1.2.8

Stylesheet Version 1.0

# UNINSTALL OF AN ATTACHED DEVICE

## Cross Reference to Related Applications

This application claims the benefit of U.S. Provisional Application No. 60/293,894, filed May 24, 2001.

## Background of Invention

[0001] The present invention relates to computers, and in particular to uninstalling a device attached to a computer.

## Summary of Invention

[0002] The invention provides a method, apparatus, system, and signal-bearing medium for uninstalling software specific to a device. Entries in configuration data for a device with an unknown type are removed. The vendor of the device with the unknown type is determined, and subkeys in the configuration data for all devices associated with the vendor are searched. Keys in the configuration data associated with the vendor are deleted. Entries associated with the device in an initialization file are removed, and selected files associated with the device are deleted. In this way, all references to a device are removed, so as not to interfere with other applications or functions.

[0003] In one particular embodiment, the invention is directed to software for uninstalling a previously installed device attached to a computer system where the previous installation was partial, incomplete, or otherwise was in error or which resulted in the operating system being unable to completely identify the attached device, or where the device is desired to be removed from the system so that a reinstallation of the device will occur without trouble or other undesired incident. The software insures that all files and artifacts that are not normally removed by the operating system are in fact removed or renamed so as not to interfere with other applications or functions, or not to interfere

with a subsequent reinstallation of the device. In a particular embodiment, the software is capable of appropriating generic registry keys and renaming manufacturer-specific registry keys so that the computer system or operating system can properly detect the device at a later time such as during a subsequent installation of the device.

## Brief Description of Drawings

[0004] Fig. 1 is a block diagram of a computer system incorporating an uninstall function, according to an embodiment of the invention.

[0005] Fig. 2 is a flowchart of processing for a reinstall of an operating system, according to an embodiment of the invention.

[0006] Fig. 3 is a flowchart of processing for an uninstall function in a stand-alone utility, according to an embodiment of the invention.

[0007] Fig. 4 is a flowchart of processing for an uninstall function during reinstallation of an operating system, according to an embodiment of the invention.

## Detailed Description

[0008] In the following detailed description of exemplary embodiments of the invention, reference is made to the accompanying drawings (where like numbers represent like elements), which form a part hereof, and in which is shown by way of illustration specific exemplary embodiments in which the invention may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention, but other embodiments may be utilized and logical, mechanical, electrical, and other changes may be made without departing from the scope of the present invention. The following detailed description is, therefore, not to be taken in a limiting sense, and the scope of the present invention is defined only by the appended claims.

[0009] In the following description, numerous specific details are set forth to provide a thorough understanding of the invention. However, it is understood that the invention may be practiced without these specific details. In other instances, well-known circuits, structures and techniques have not been shown in detail in order not to obscure the invention.

[0010] Fig. 1 depicts a block diagram of data processing system 100, according to an

embodiment of the invention. Data processing system 100 includes computer 120 attached to network 130 and attached device 175. In another embodiment, network 130 is not present.

[0011] Computer 120 includes processor 135, storage device 140, network adapter 145, input device 150, output device 155, and bus adapter 174, all communicatively coupled via system bus 180.

[0012] Processor 135 represents a central processing unit of any type of architecture, such as a CISC (Complex Instruction Set Computing), RISC (Reduced Instruction Set Computing), VLIW (Very Long Instruction Word), or a hybrid architecture, although any appropriate processor may be used. Processor 135 executes instructions and includes that portion of computer 120 that controls the operation of the entire computer. Although not depicted in Fig. 1, processor 135 typically includes a control unit that organizes data and program storage in computer memory and transfers data and other information between the various parts of computer 120. Processor 135 may receive input data from input device 150 and network adapter 145, read and store code and data in storage device 140, and may present output data to a user via output device 155. Processor 135 also may transmit and receive packets of information across network 130 using network adapter 145.

[0013] Although computer 120 is shown to contain only a single processor and a single bus, the present invention applies equally to computers that may have multiple processors and to computers that may have multiple buses with some or all performing different functions in different ways.

[0014] Storage device 140 represents one or more mechanisms for storing data. For example, storage device 140 may include read only memory (ROM), random access memory (RAM), magnetic disk storage media, optical storage mediums, flash memory devices, and/or other machine-readable media. Although only one storage device 140 is shown, multiple storage devices and multiple types of storage devices may be present. Further, although computer 120 is drawn to contain storage device 140, in other embodiments storage device 140 may be distributed across other computers attached via a network, such as network 130.

[0015] Storage device 140 contains uninstall controller 160, configuration data 165,

operating system 170, initialization file 172, and file list 173. Uninstall controller 160 may include instructions capable of being executed on processor 135 to carry out the functions of the present invention, as further described below with reference to Figs. 3 and 4. In another embodiment, some or all of the functions of the present invention may be carried out via hardware in lieu of a processor-based system. Of course, storage device 140 may also contain additional software and data (not shown), which is not necessary to understanding the invention.

[0016] In an embodiment, operating system 170 may be the Microsoft Windows operating system. But in other embodiments, operating system 170 may be the Linux, Unix, Macintosh, or any other appropriate operating system.

[0017] Configuration data 165 stores information necessary to configure computer system 100. In one embodiment, configuration data 165 may be a hierarchical database, but in other embodiments configuration data 165 may be any type of data repository. Although configuration data 165 is shown as being a single unit, in other embodiments, configuration data 165 is composed of multiple pieces of data in multiple locations. Configuration data 165 contains information that operating system 170 references during operation. In an embodiment, configuration data 165 may be the registry of the Windows operating system. The Windows registry replaces the files "autoexec.bat" and "config.sys" in DOS (Disk Operating System). But, in other embodiments, configuration data 165 may be any appropriate configuration data. Configuration data 165 contains keys 166. Keys 166 may contain settings information for values in configuration data 165. Keys 166 are further described below with reference to Fig. 3.

[0018] Initialization file 172 contains data that is processed by operating system 170 when the computer boots up. In an embodiment, initialization file 172 may be the Windows "Win.ini file," but in other embodiments, initialization file 172 may be any appropriate initialization file. File list 173 contains a list of files to be deleted and is further described below with respect to Fig. 3.

[0019] Referring again to Fig. 1, system bus 180 represents one or more busses (e.g., PCI, ISA (Industry Standard Architecture), X-Bus, EISA (Extended Industry Standard Architecture), or any other appropriate bus) and bridges (also termed bus controllers).

[0020] Input device 150 is that part of computer 120 that accepts input from a user. In an

embodiment, input device 150 may be a keyboard, but in other embodiments, input device 150 may be a pointing device, mouse, trackball, keypad, touchpad, touch screen, pointing stick, microphone, or any other appropriate input device. Although only one input device 150 is shown, in other embodiments any number of input devices of the same or of a variety of types may be present.

[0021] Output device 155 communicates information to the user of computer 120. Output device 155 may be a cathode-ray tube (CRT) based video display well known in the art of computer hardware. But, in other embodiments output device 155 may be replaced with a liquid crystal display (LCD) based or gas, plasma-based, flat-panel display. In still other embodiments, any appropriate display device is used. In yet embodiments, a speaker that produces audio output may be used. Although only one output device 155 is shown, in other embodiments, any number of output devices of different types or of the same type may be present. In other embodiments, output device 155 might not be present.

[0022] Network adapter 145 facilitates communication between computer 120 and network 130. Network adapter 145 provides a user of computer 120 with a means of electronically communicating information, such as code and data, with an unillustrated computer or computers attached to network 130. In addition, in another embodiment, network adapter 145 may support distributed processing, which enables computer 120 to share a task with other devices linked to network 130. Although network adapter 145 is shown as part of computer 120, in another embodiment they may be packaged separately. Although only one network adapter 145 is shown, in other embodiments, multiple network adapters of the same or of a variety of types may be present.

[0023] Bus adapter 174 sends and receives data and commands between system bus 180 and device bus 176. In an embodiment, device bus 176 may be a USB (Universal Serial Bus). In other embodiments, device bus 176 may be a IEEE 1394 bus, a wireless interface such as IEEE 802.11, or BlueTooth. In still other embodiments any appropriate serial, parallel, or other type of bus may be used.

[0024] In an embodiment, attached device 175, which is attached to computer 120 via device bus 176, may be a printer. But, in other embodiments attached device may be a storage device, hard disk, floppy disk, semiconductor memory device, CD drive, DVD drive, device reader, keyboard, display device, modem, scanner, network adapter, mouse, touchpad, joystick, foot pedal, camera, pointing device, microphone, or any other kind of

memory, storage, input device, or output device capable of being attached and installed to computer 120 and subsequently uninstalled. Although attached device 175 is shown as being external to computer 120, in other embodiments attached device 175 may be attached internally to computer 120. In still other embodiments, attached device 175 may be attached indirectly to computer 120 via a network, such as network 130.

[0025] Computer 120 may be implemented using any suitable hardware and/or software, such as a personal computer available from a number of vendors. Portable computers, laptop or notebook computers, mainframe computers, handheld devices, PDAs (Personal Digital Assistants), and network computers or Internet appliances are examples of other possible configurations. The hardware and software depicted in Fig. 1 may vary for specific applications and may include more or fewer elements than those depicted. For example, other peripheral devices such as audio adapters, or chip programming devices, such as EPROM (Erasable Programmable Read-Only Memory) programming devices may be used in addition to or in place of the hardware already depicted. Thus, an embodiment of the invention may apply to any hardware configuration that supports the installation and uninstallation of attached devices.

[0026] Network 130 may be any suitable network. Although only one network 130 is shown, in other embodiments any number of networks may be present. In an embodiment, network 130 may support wireless communications. In another embodiment, network 130 may support hard-wired communications, such as a telephone line or cable. Network 130 may support any appropriate protocol. In an embodiment networks 130 is the Internet and supports TCP/IP (Transmission Control Protocol/Internet Protocol). In other embodiments, network 130 may be a local area network (LAN) or a wide area network (WAN).

[0027] As will be described in detail below, aspects of an embodiment pertain to specific apparatus and method elements implementable on computers. In another embodiment, the invention may be implemented as a program product for use with a computer. The programs defining the functions of this embodiment may be delivered to a computer via a variety of signal-bearing media, which include, but are not limited to:

[0028] (1) information permanently stored on non-rewriteable storage media (e.g., read only memory devices within a computer such as CD-ROM readable by an unillustrated CD-ROM drive;

[0029] (2) alterable information stored on rewriteable storage media (e.g., a hard disk drive or diskette); or

[0030] (3) information conveyed to a computer by a communications medium, such as through a computer or telephone network accessed via network adapter 145, including wireless communications.

[0031] Such signal-bearing media, when carrying processor-readable instructions that direct the functions of the present invention, represent embodiments of the present invention.

[0032] Fig. 2 is a flowchart of processing for a reinstall of an operating system on a computer, according to an embodiment of the invention. Although Fig. 2 will be described using Microsoft Windows terminology, the functions of Fig. 2 apply equally to any appropriate operating system. Control begins a block 200. Control then continues to block 205 where during a reinstall of the Microsoft Windows operating system, one of the last actions of the Windows Setup function is to detect an attached device if it is plugged in. Windows attempts to determine what type of a device is attached to a port and whether or not there are drivers available to install for this device.

[0033] Control then continues to block 210 where Windows identifies the attached device as an "Unknown Device" because the drivers for the attached device are not yet installed. Thus, attached device 175 may be identified by system 100 as an "unknown device" or "unknown type." "Unknown" as defined herein means that the device is not identified or is only partially identified by operating system 170. One particular definition of "unknown" as set forth herein means that the drivers for the device have not been installed, have been incompletely installed, or are not the better drivers compared with other alternative drivers available to be installed. For example, an unknown device may include a device on a system that has generic drivers installed, but does not have manufacturer-specific drivers installed, or does not have later versions of the drivers installed. As defined herein, an attached device may be specified as an "unknown device" or of an "unknown type" by operating system 170 while simultaneously being known and identified by a user, the manufacturer, another operating system, etc. A device with an unknown type is an unknown device.

[0034] Control then continues to block 215 where Windows completes its installation and installs applications and device drivers for recognized devices. Control then continues to

block 220 where the functions of uninstall controller 160 are invoked, as further described below with reference to Fig. 4. Referring again to Fig. 2, control then continues to block 225 where the appropriate driver is installed from an installation medium, such as a CD.

[0035] Control then continues to block 230 where the computer is rebooted, and Windows redetects the attached device. Control then continues to block 235 where Windows associates the installed device driver, which is now present, with the detected attached device. Control then continues to block 299 where the process ends.

[0036] The flowchart of Fig. 3 describes the processing for functions of uninstall controller 160 in a stand-alone utility, which searches through configuration data 165 (the Windows registry in one embodiment) in the area dealing with the attached devices of computer 120 and detects whether or not an attached device, such as attached device 175, has been installed. Uninstall controller 160 then eliminates files and registry entries related to that attached device. After this process is completed, the attached device may be installed either from the device vendor's installation medium or from the drivers on the operating system's installation medium (The Windows Restoration CD in an embodiment).

[0037] Control begins at block 300. Control then continues to block 305 where uninstall controller 160 searches configuration data 165 for any devices that have a description of "Unknown" and removes them regardless of what type of device it really is. In a Windows embodiment, uninstall controller searches HKEY_LOCAL_MACHINE\Enum\USB.

[0038] Control then continues to block 310 where uninstall controller 160 rescans subkeys within keys 166 for any data related to an attached device. In a Windows embodiment, uninstall controller 160 scans the HKEY_LOCAL_MACHINE\Enum\USB registry subkey for any data related to an attached device. The Windows registry file is organized in a hierarchical fashion, and a subkey is a key within a key, analogous to a subdirectory within a directory. The device vendors identify their devices in different ways, so uninstall controller 160 detects which vendor has its attached device installed and only uninstalls that attached device vendor's drivers. This saves time and allows for more than one attached device of the same vendor to be removed.

[0039] Once an attached device has been found under the

HKEY_LOCAL_MACHINE\Enum\USB subkey (in a Windows embodiment) and the vendor has been determined, control continues to block 315 where uninstall controller 160 removes the appropriate registry keys. Below is attached a list of common registry keys that are searched and removed in a Windows embodiment if there are attached device references in them. The below registry keys are examples for a USB printer device, but any type of device and bus may be used in other embodiments.

[0040] HKEY_LOCAL_MACHINE\enum\usb\

[0041] HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Print\Environments\Windows 4.0\Drivers

[0042] HKEY_LOCAL_MACHINE\System\currentcontrolset\control\print\monitors

[0043] HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Print\Ports

[0044] HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Print\Printers

[0045] HKEY_LOCAL_MACHINE\Config\0001 \System\CurrentControlSet\Control\Print\Printers

[0046] HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Class

[0047] More generically, for any attached device 175, any one or more of the following registry keys are searched and removed:

[0048] HKEY_LOCAL_MACHINE\enum\device bus\

[0049] HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\device function\Environments\Windows 4.0\Drivers

[0050] HKEY_LOCAL_MACHINE\System\currentcontrolset\control\device function\monitors

[0051] HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\device function\Ports

[0052] HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\device function\device type

[0053] HKEY_LOCAL_MACHINE\Config\0001\System\CurrentControlSet\Control\device

function\device type

[0054] HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Class, where device bus indicates a type of bus the predetermined device uses to couple to the computer system (e.g., USB), device function indicates a function the predetermined devices provides (e.g., print), and device type indicates a group classification of the predetermined device (e.g., printer).

[0055] Control then continues to block 320 where uninstall controller 160 modifies initialization file 172. In a Windows embodiment, initialization file 172 may be the "Win.ini" file, and uninstall controller 160 may clear the "load=", "run=", "device=" lines, and the "devices" and "printerports" sections. This will help operating system 170 from assuming that attached devices are installed. Generically, "printerports" can be described herein as "deviceports" where device refers to the type of the device.

[0056] Control then continues to block 330 where uninstall controller 160 deletes files related to the attached device. In the windows embodiment, the files may be in "windows/inf," "windows/system," and "windows\system32." Uninstall controller 160 retrieves a list of files to search for from file list 173. When these files are found, a copy is made of these files adding a ".bkp" extension to the end. Windows will not know what file it actually is, but technicians using uninstall controller 160 may get these files back. In a windows embodiment, uninstall controller 160 adds the original file to the "windows\wininit.ini" file. Windows will either rename or delete the files listed in this file depending on the entry. Uninstall utility 160 informs Windows to delete the files because there was already a back up made earlier.

[0057] Control then continues to block 340 where uninstall controller 160 restarts computer 120 in order for the "windows\wininit.ini" file to be processed, in a Windows embodiment. Computer 120 is now in a similar state as the point before attached device 175 was plugged in, and attached device 175 may now be reinstalled.

[0058] Control then continues to block 399 where the function returns.

[0059] Fig. 4 is a flowchart of processing for an uninstall function during reinstallation of an operating system, according to an embodiment of the invention.

[0060] Control begins at block 400. Control then continues to block 405 where uninstall

controller 160 searches configuration data 165 for any devices that have a description of "Unknown" and removes them regardless of what type of device it really is. In a Windows embodiment, uninstall controller searches HKEY_LOCAL_MACHINE\Enum\USB.

[0061] Control then continues to block 410 where uninstall controller 160 rescans subkeys within keys 166 for any data related to an attached device. In a Windows embodiment, uninstall controller 160 scans the HKEY_LOCAL_MACHINE\Enum\USB registry subkey for any data related to an attached device. The Windows registry file is organized in a hierarchical fashion, and a subkey is a key within a key, analogous to a subdirectory within a directory. The device vendors identify their devices in different ways, so uninstall controller 160 detects which vendor has its attached device installed and only uninstalls that attached device vendor's drivers. This saves time and allows for more than one attached device of the same vendor to be removed.

[0062] Once an attached device has been found under the HKEY_LOCAL_MACHINE\Enum\USB subkey (in a Windows embodiment) and the vendor has been determined, control continues to block 415 where uninstall controller 160 removes the appropriate registry keys. Below is attached a list of common registry keys that are searched and removed in a Windows embodiment if there are attached device references in them. The below registry keys are examples for a USB printer device, but any type of device and bus may be used in other embodiments.

[0063] HKEY_LOCAL_MACHINE\enum\usb\

[0064] HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Print\Environments\Windows 4.0\Drivers

[0065] HKEY_LOCAL_MACHINE\System\currentcontrolset\control\print\monitors

[0066] HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Print\Ports

[0067] HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Print\Printers

[0068] HKEY_LOCAL_MACHINE\Config\0001 \System\CurrentControlSet\Control\Print\Printers

[0069] HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Class

[0070] More generically, for any attached device 175, any one or more of the following registry keys are searched and removed:

[0071] HKEY_LOCAL_MACHINE\enum\device bus\

[0072] HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\device function\Environments\Windows 4.0\Drivers

[0073] HKEY_LOCAL_MACHINE\System\currentcontrolset\control\device function\monitors

[0074] HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\device function\Ports

[0075] HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\device function\device type

[0076] HKEY_LOCAL_MACHINE\Config\0001\System\CurrentControlSet\Control\device function\device type

[0077] HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Class, where device bus indicates a type of bus the predetermined device uses to couple to the computer system (e.g., USB), device function indicates a function the predetermined devices provides (e.g., print), and device type indicates a group classification of the predetermined device (e.g., printer).

[0078] Control then continues to block 499 where the function returns.

[0079] The invention described herein removes all traces of an attached device 175 that has been installed on system 100. For example, where attached device 175 is a USB printer, and bus adapter 174 and device bus 176 are a USB bus adapter and USB device bus, respectively, the registry of system 100 and other places in system 100 are searched for files and entries related to USB printers are removed, so that all traces of the USB printer are removed from system 100. For example, the invention searches the registry to determine what kinds of attached device 175 have been installed or have been attempted to be installed. Registry entries for a predetermined one of attached device 175 are removed, and also any Win.ini files associated with the predetermined attached device 175. As a result, system 100 will contain no traces of the previous installation of the USB printer so that system is "clean", meaning that no files or other remnants of that particular USB device remain in system 100. In the event of a future reinstallation of the

particular device, such as during a system restoration, attached device 175 will be correctly identified and the proper drivers will be correctly loaded.

[0080] Although the invention has been described with a certain degree of particularity, it should be recognized that elements thereof may be altered by persons skilled in the art without departing from the spirit and scope of the invention and without providing a substantial change thereto. One of the embodiments of the invention can be implemented as sets of instructions resident in the memory of one or more computer systems configured generally as described in Fig. 1. Until required by the computer system, the set of instructions may be stored in another computer readable memory such as an auxiliary memory, for example in a hard disk drive or in a removable memory such as an optical disk for utilization in a CD-ROM drive, a floppy disk for utilization in a floppy disk drive, a combination floppy and optical disk for utilization in a floppy-optical drive, or a personal computer memory card for utilization in a personal computer card slot. Further, the set of instructions can be stored in the memory of another computer and transmitted over a local area network or a wide area network, such as the Internet, when desired by the user. Additionally, the instructions may be transmitted over a network in the form of an applet or a servlet that is interpreted or compiled after transmission to the computer system rather than prior to transmission. One skilled in the art would appreciate that the physical storage of the sets of instructions or applets physically changes the medium upon which it is stored electrically, magnetically, chemically, physically, optically or holographically so that the medium carries computer readable information.